

Package ‘poio’

October 14, 2022

Title Input/Output Functionality for ``PO" and ``POT" Message Translation Files

Version 0.0-4

Maintainer Richard Cotton <richierocks@gmail.com>

URL <https://github.com/RL10N/poio>

BugReports <https://github.com/RL10N/poio/issues>

Description Read and write PO and POT files, for package translations.

Depends R (>= 3.2.5)

Imports assertive.base, assertive.properties, assertive.types, assertive.files, assertive.sets, assertive.strings, devtools, digest, dplyr, magrittr, purrr, R6, stringi, tibble, tools, utils, whoami

Suggests testthat, rebus.datetimes

License GPL-3

Acknowledgements Development of this package has been funded by the R Consortium (<https://r-consortium.org>).

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Richard Cotton [aut, cre],
Thomas Leeper [aut]

Repository CRAN

Date/Publication 2020-04-17 10:30:02 UTC

R topics documented:

fix_metadata	2
generate_po_from_pot	4

get_n_plural_forms	5
language_codes	5
plural_forms	6
po	7
poio	9
print.po	9
read_po	10
summary.po	11
write_po	11

Index	13
--------------	-----------

fix_metadata	<i>Fix metadata in a PO object</i>
--------------	------------------------------------

Description

Fixes the metadata in a po object, as generated by [read_po](#).

Usage

```
fix_metadata(x, pkg = ".", ..., .dots = list())

## S3 method for class 'po'
fix_metadata(
  x,
  pkg = ".",
  clone = TRUE,
  file_type = x$file_type,
  ...,
  .dots = list()
)

## S3 method for class 'data.frame'
fix_metadata(x, pkg = ".", file_type, ..., .dots = list())
```

Arguments

x	An object of class po, or the data frame from the metadata element of such an object.
pkg	A path to the root of an R package source directory, or a package object, as created by as.package .
...	Named arguments of new metadata values.
.dots	A named list of new metadata values.
clone	Logical. If TRUE, the po object is cloned before the metadata is fixed. This has a slight performance cost, but is easier to reason about.
file_type	A string giving the file type; either "po" or "pot".

Details

Columns are added to ensure that the metadata data frame contains character columns named "name" and "value". Likewise rows are added or updated as follows.

Project-Id-Version The package name and version, taken from the "Package" and "Version" fields of the DESCRIPTION file specified in the pkg argument.

Report-Msgid-Bugs-To The URL to report bugs to, taken from the "BugReports" field of the DESCRIPTION file specified in the pkg argument.

POT-Creation-Date Not auto-updated.

PO-Revision-Date The current date and time, in format " and time formatting specifications.

Last-Translator Your name and email, creepily autodetected by [whoami](#), where possible.

Language-Team Not auto-updated. Invent your own team name!

MIME-Version Always changed to "1.0".

Content-Type Always changed to "text/plain; charset=UTF-8".

Content-Transfer-Encoding Always changed to "8bit".

Additionally PO, but not POT, files have these rows:

Language An ISO 639-1 two-letter language code. See http://www.loc.gov/standards/iso639-2/php/code_list.php

Plural-Forms The plural-form specification for the Language code.

Value

An object of the same type as the input, but with the metadata fixed.

Examples

```
pot_file <- system.file("extdata/R-summerof69.pot", package = "poio")
pot <- read_po(pot_file)
pot_fixed <- fix_metadata(pot, system.file(package = "poio"))

# Choose your own metadata
pot_fixed2 <- fix_metadata(
  pot,
  system.file(package = "poio"),
  "Last-Translator" = "Dr. Daniel Jackson <djackson@stargate.com>",
  .dots = list(
    "Language-Team" = "Team RL10N!"
  )
)

# Compare the metadata before and after
pot$metadata
pot_fixed$metadata
```

generate_po_from_pot *Generate a PO object from a POT object*

Description

Generates a PO object from a POT object.

Usage

```
generate_po_from_pot(x, lang, ...)
```

```
## S3 method for class 'po'  
generate_po_from_pot(x, lang, ...)
```

Arguments

x	An object of class po, as read by read_po from a POT file.
lang	A language code, possibly with a country code attached. See language_codes for possible values.
...	Currently unused.

Details

The `file_type` element is changed from "pot" to "po", and "Language" and "Plural-Forms" values are added to the metadata element.

Value

An object of class po, ready to be written to a PO file by [write_po](#).

Note

If the plural form is unknown for the specified language, the plural form is set to NA. See [plural_forms](#) for supported languages.

Examples

```
pot_file <- system.file("extdata/R-summerof69.pot", package = "poio")  
pot <- read_po(pot_file)  
# It's a good idea to fix the metadata before you generate the PO files  
pot_fixed <- fix_metadata(pot, system.file(package = "poio"))  
  
# Call generate_po_from_pot for each language that you want to translate to  
two_pos <- lapply(  
  c(German = "de", Qatari_Arabic = "ar_QA"),  
  generate_po_from_pot,  
  x = pot_fixed  
)
```

```
# Notice the Language and Plural-forms elements in the metadata
two_pos$German$metadata
# Also notice that the countable msgstr elements for Arabic now
# have length 6, since Arabic has 6 plural forms
two_pos$Qatari_Arabic$countable$msgstr
```

get_n_plural_forms *Get the number of plural forms from a PO file*

Description

Gets the number of plural forms specified in the "Plural-Forms" metadata element of a PO file.

Usage

```
get_n_plural_forms(x, default = 2L, ...)
```

Arguments

x	A po object or its metadata element.
default	An integer to return if the number of plural forms cannot be determined.
...	Arguments passed between methods.

Value

An integer of the number of plural forms for the language defined in the PO file.

Note

POT files are not language-specific and don't have a "Plural-Forms" metadata element. By convention, they are considered to have 2 plural forms, since that is how many plural forms there are in English.

language_codes *Language codes supported by GNU gettext*

Description

This dataset contains the language and country code values that are accepted by GNU gettext. Its primary purpose is to allow checking of the "Language" field of PO translation files.

Usage

```
data(language_codes)

ALLOWED_LANGUAGE_REGEX
```

Format

language_codes is a list with two character vector elements.

language Lowercase two letter ISO 639-1 codes, and some lowercase three letter ISO 639-2 codes representing languages.

country Uppercase two letter ISO 3166-1 alpha-2 code representing countries and territories.

An object of class character of length 1.

Details

Valid "Language" field values consist of a language code taken from the language element of this dataset, optionally followed by an underscore and a country code taken from the country element of this dataset.

Note

The language element of the dataset contains all of ISO 639-1 as well as the value "mo", for "Moldavian", which isn't an official ISO code. It also contains a subset of ISO 639-2, for rare languages where there is no ISO 629-1 code. The country element is identical to ISO 3166-1 alpha-2.

References

The dataset was generated from these the contents of these webpages: https://www.gnu.org/software/gettext/manual/html_node/Usual-Language-Codes.html#Usual-Language-Codes https://www.gnu.org/software/gettext/manual/html_node/Rare-Language-Codes.html#Rare-Language-Codes https://www.gnu.org/software/gettext/manual/html_node/Country-Codes.html#Country-Codes

Examples

```
# The dataset contains:
e <- new.env()
utils::data(language_codes, package = "poio", envir = e)
e$language_codes

# Allowed values in the language field can be matched like this
# (though it will be automatically done in generate_po_from_pot)
stringi::stri_detect_regex(c("it", "nl_BE", "xxx"), ALLOWED_LANGUAGE_REGEX)
```

plural_forms

Plural Forms Data

Description

This dataset contains the known values for the "Plural-Forms" metadata elements in a PO file.

Usage

```
data(plural_forms)
```

Format

`plural_forms` is a data frame with 3 character columns and 1 numeric column and 142 rows:

ISO The ISO 639 specification of the language. Usually this is the two letter ISO 639-1 name, but some three letter ISO 639-2 names are also included, for cases where no two letter name exists.

EnglishName How the language is known in English.

PluralFormHeader The GNU gettext specification of how plural forms are specified in the language.

nplurals A numeric value specifying the number of plural forms.

References

This dataset was originally taken from <http://localization-guide.readthedocs.org/en/latest/l10n/pluralforms.html>

ISO 639-1 (a.k.a. ISO 629-2 alpha-2) and ISO 639-2 language codes can be found in the [ISO_639_2](http://www.loc.gov/standards/iso639-2/php/code_list.php) help page, or online at http://www.loc.gov/standards/iso639-2/php/code_list.php

Examples

```
e <- new.env()
data(plural_forms, package = "poio", envir = e)
e$plural_forms
```

po *Create an object of class po*

Description

Creates an object of class `po`, for storing package translations.

Usage

```
po(source_type, file_type, initial_comments, metadata, direct, countable)
```

Arguments

<code>source_type</code>	Either "r" or "c", depending upon whether the messages originated from R-level code, or C-level code.
<code>file_type</code>	Either "po" or "pot", depending upon whether the messages originated from a PO (language-specific) or POT (master translation) file.

<code>initial_comments</code>	A character vector of comments added by the translator.
<code>metadata</code>	A <code>tibble</code> of file metadata with columns "name" and "value".
<code>direct</code>	A <code>tibble</code> of messages with a direct translation, as created by <code>stop</code> , <code>warning</code> , <code>message</code> or <code>gettext</code> ; its columns are described below.
<code>countable</code>	A data frame of messages where the translation depends upon a countable value, as created by <code>ngettext</code> ; its columns are described below.

Value

An R6 object of class `po`.

Note

#' The `direct` element of the `po` object has the following columns.

msgid Character. The untranslated (should be American English) message.

msgstr Character. The translated message, or empty strings in the case of POT files.

is_obsolete Logical. Is the message obsolete?

msgctxt List of character. Disambiguating context information to allow multiple messages with the same ID.

translator_comments List of character. Comments added by the translator, typically to explain unclear messages, or why translation choices were made.

source_reference_comments List of character. Links to where the message occurred in the source, in the form "filename:line".

flags_comments List of character. Typically used to describe formatting directives. R uses C-style formatting, which would imply a "c-format" flag. For example string. "fuzzy" flags can appear when PO files are merged.

previous_string_comments List of character. When PO files are merged with an updated POT file, and a fuzzy flag is generated, the old `msgid` is stored in a previous string comment.

The `countable` element of the `po` object takes the same form as the `direct` element, with two differences.

msgid_plural Character. The plural form of the untranslated message.

msgstr This is now a list of character (rather than character.)

The `direct` and `countable` elements also have a read-only column named `msgkey` that acts as a key for the message. It is generated with `digest` and `algo = "xxhash32"` on the `msgid` and `msgctxt` fields.

References

Much of the logic for this function was determined from reading http://pology.nedohodnik.net/doc/user/en_US/ch-poformat.html

poio	<i>Input/Output Functionality for “PO” and “POT” Message Translation Files</i>
------	--

Description

Read and write PO and POT files, for package translations.

Author(s)

Richard Cotton

References

[GNU gettext Manual](#)

See Also

[read_po](#), [write_po](#)

print.po	<i>Print a PO object</i>
----------	--------------------------

Description

[print](#) method for PO objects.

Usage

```
## S3 method for class 'po'  
print(x, ...)
```

Arguments

x	An object of class po.
...	Arguments passed to print.tbl_df .

Value

The x argument is invisibly returned, but the function is mostly invoked for the side-effect of printing the x argument.

See Also

[summary.po](#)

Examples

```
pot_file <- system.file("extdata/R-summerof69.pot", package = "poio")
print(pot <- read_po(pot_file))

# Use width = Inf to print all columns in metadata, direct, and countable
print(pot, width = Inf)
```

read_po	<i>Read PO and POT files</i>
---------	------------------------------

Description

Reads .PO and .POT translation files.

Usage

```
read_po(po_file)
```

Arguments

po_file A string giving a path to a PO file.

Value

An object of class `po`. The `source_type` and `file_type` elements are automatically determined from the file name.

See Also

[xgettext](#)

Examples

```
# read_po is used for both po and pot files
pot_file <- system.file("extdata/R-summerof69.pot", package = "poio")
(pot <- read_po(pot_file))
```

summary.po	<i>Summarize a PO object</i>
------------	------------------------------

Description

[summary](#) method for PO objects. `summary` shows less information than `print`.

Usage

```
## S3 method for class 'po'  
summary(object, ...)
```

Arguments

<code>object</code>	An object of class <code>po</code> .
<code>...</code>	Arguments passed to print.data.frame .

Value

The object argument is invisibly returned, but the function is mostly invoked for the side-effect of printing the object argument.

See Also

[print.po](#)

Examples

```
pot_file <- system.file("extdata/R-summerof69.pot", package = "poio")  
summary(pot <- read_po(pot_file))
```

write_po	<i>Write a PO file</i>
----------	------------------------

Description

Writes an object of class `po` to a `.po` file.

Usage

```
write_po(po, po_file = NULL, ...)
```

Arguments

<code>po</code>	An object of class <code>po</code> .
<code>po_file</code>	A path to the <code>po_file</code> to be written, or <code>NULL</code> to automatically generate the path.
<code>...</code>	Passed between methods. Not currently used.

Value

The function is mostly invoked for the side-effect of writing a PO file. The `po` argument is also invisibly returned, for convenience when this function is used in a pipe chain.

Examples

```
pot_file <- system.file("extdata/R-summerof69.pot", package = "poio")
pot <- read_po(pot_file)
write_po(pot, stdout())
```

Index

- * **datasets**
 - language_codes, 5
 - plural_forms, 6
- * **package**
 - poio, 9
- ALLOWED_LANGUAGE_REGEX
 - (language_codes), 5
- as.package, 2
- digest, 8
- fix_metadata, 2
- generate_po_from_pot, 4
- get_n_plural_forms, 5
- gettext, 8
- ISO_639_2, 7
- language_codes, 4, 5
- message, 8
- ngettext, 8
- plural_forms, 4, 6
- po, 7, 10
- poio, 9
- print, 9
- print.data.frame, 11
- print.po, 9, 11
- print.tbl_df, 9
- R6, 8
- read_po, 2, 4, 9, 10
- stop, 8
- summary, 11
- summary.po, 9, 11
- tibble, 8
- warning, 8
- whoami, 3
- write_po, 4, 9, 11
- xgettext, 10